Feedback from previous attempt. It was too long to do a screenshot

Unfortunately, the program crashed when the user selects the "display" menu option before creating a customer. The issue is caused by the "sCurrentCustomer" variable, which is not needed for this program. The solution is to rework the program's menu handling until everything is working through the current customer variable.

Your grade before penalties would have been 47%.

Structure/Efficiency: Poor 0/10,

Good work but the CheckingAccount's variables should have been declared as private. Remember to continually practice data encapsulation by preventing direct access to a class' data. Unless there is a specific reason not to, all variables should be declared as private with public getters & setters if need be.

The "GetAccountInformation" method accepts what appears to be what the method is supposed to be returning. It is illogical to expect a developer to pass data to a method when they are trying retrieve that data. This method should be removed since individual attributes of a class are expected to be accessed individually. If this was to be used to access all the data for display purposes, overide the ToString method to concatenate all the class' members into a single string value.

The default constructor for the "Customer" class was not needed as we never want a customer object created without a name.

Good work setting up the classes; however, mutator (set) methods shouldn't be implemented unless specifically needed. For example, the customer's name and the checking account's number won't change after the customer object has been instantiated. Only the customer's checking account object and the checking account balance will be mutated (set) after the objects have been instantiated.

Per the previous feedback, "The Main method has a lot of extra variables declared at the top of it that aren't really needed. The application only needs to store all of the information for one customer at any given time. The goal is to allow for one Customer object to store all of that data rather than have several separate variables all containing that information. If the application were to be expanded to store information about multiple customers then it would quickly become confusing with several variables being stored for each customer rather than having one Customer object per customer and each object stores all of the information for one specific customer."

The if-statements for the user selections were all individual statements. Since the user can only select one menu option at a time all of the if-statements related to user selections should have been conjoined using the "else-if" statement. Doing so would save processing time / power by skipping the remaining if-statements as soon as one of them resolved to true. This is automatically performed by a switch-statement. A switch-statement is preferred when each criteria check is an equivalency between a series of hard-coded / literal values. If-statements are needed for greater / less than comparisons or when dynamic comparisons need to be performed.

The goal customer object was used directly in string concatenation. By default the class name and the package which contains it, is what is concatenated into the string. Implement an overridden "ToString" method in the customer class to change the default string value.

The "Validation" class was present and used but other instances of manual input validation was present as well. Since that helper / utility class exists it makes sense to use it for all user input instead of intermittently.

Classes: Fair 4.5/15,

Unfortunately, the variable which was to represent the customer's checking account is missing. A class can be used as a data type. As such, any new class we write can be used as a data type for a variable. In this case, we were supposed to use our checking account class as the data type for the customer's checking account member.

Constructors: Poor 0/15,

Unfortunately, the "CheckingAccount" constructor did not accept the arguments which were detailed in the instructions. Review the instructions and ask questions if there is anything unclear about how to add parameters to a constructor method.The "Customer" constructor was to accept a String parameter for its name, but did not need any other information beyond that. The "account" related information was to be collected from the user and used to create a checking account for the customer in the "create a checking account" menu option. It is expected that a customer may exist without a checking account.

Menu: Fair 7.5/25,

The "create account" menu option creates a new customer object. This menu option was to be used to instantiate a checking account class object then assign it to the current customer (only if the customer was created by the user first).

Likewise, the "set balance" menu option created a new customer. Imagine if your friend walked into a bank to deposit a check and a completely different person came out claiming to be your friend. Just because to people have the same name and same account info does not make them the same person. As the GoToTraining session visually illustrated, a brand new object instance is created in memory when the "new" keyword is used. The goal was to work with and through the same customer class object which the user last created. Again, if the user has not yet created a customer this menu option should be written to handle that case gracefully. A gracefully handling of that situation would be to display a message to the user informing them why the menu option could not be completed and what steps to take to resolve the issue.

The "display" method displays individual values that are not stored in the customer object. The goal for this menu option was to display the attributes stored in the current customer and any of its nested objects. Implementing an overridden "ToString" method in custom classes can make the menu code for displaying the class much cleaner / simpler.

Main: Excellent 15/15,

The grade deduction for the following comment has been applied to the "structure & efficiency" category. The variable "cust" was assigned a new instance of a Customer class object when it was declared. The goal was to build a program which could handle a variable that could be null or assigned an instance of an object. Plus, the menu system's "create customer" option indicates that the user is in control of when a customer object is created. A new customer with no name serves no purpose other than to avoid writing code to handle null reference variables. Null reference variables are part of programming, it would be better to learn how to perform "null checks / protection" now.

Input Validation: Excellent 20/20,

Don't forget to always inform the user that their input is invalid and why it was rejected. The user may become confused as to why the program is not responding when they enter something. The user is more likely to blame the program as being broken than to realize they are making a mistake.